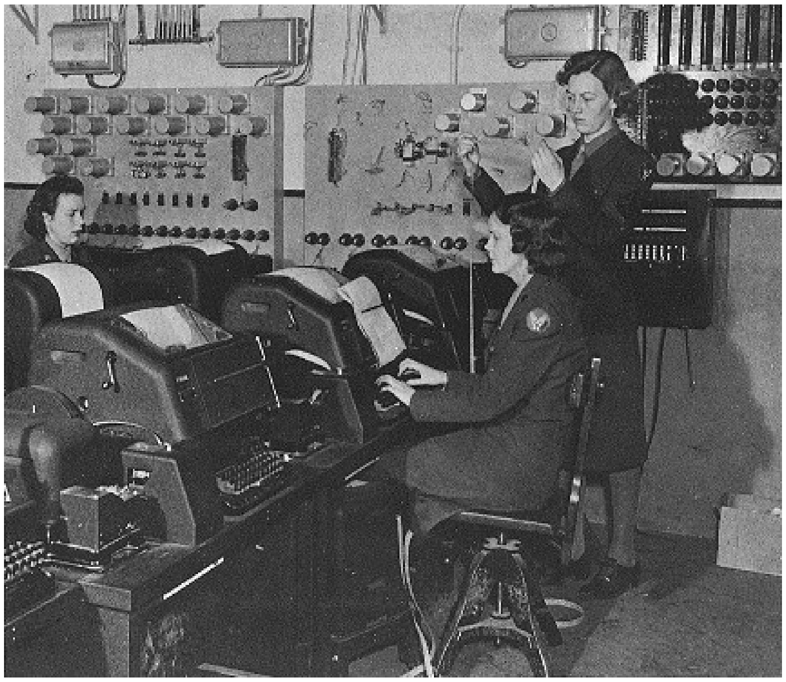
# computer literacy, coding and freedom for beginners

the concepts you need to understand modern computers are many decades old. in the early 1970s, and especially the 1960s, you were just as likely to use a teletype as a computer screen.



public domain, wikimedia commons

as the picture shows, these machines had no screen but used paper instead. the information from the typewriter was sent electronically over phone lines, and the data received could be typed automatically onto the paper as well. thus it was possible for two people to communicate over distances by electronic typewriter.

work on automating sent and received text went as far back as thomas edison, but in the 20th century these machines would be used to communicate with computers. the paper teletype eventually gave way to similar devices with video screens, having little or no processing capability of their own except to send and receive text over the same connection as a teletype. computers the size of your cabinets under and around the kitchen sink would host multiple users over telephone lines.

the point of this history lesson is to ground you in the fundamentals of computing. we could have started with beads on a string, but we skipped "ahead" to the middle of the 20th century when video screens slowly began to replace rolls of paper.

if you started writing programs on microcomputers in the 1980s, your world was not that different from the teletypes still in use in the 1970s. you had a screen in the shape of a flattened black bubble made of glass, which we will represent here with a large rectangle:

```
hello
```

in terms of user experience, the first commercially available graphical systems were from xerox. for home users, the apple macintosh gave most of the public their first introduction to icons and application windows. you can even boot a modern mac laptop into "single user mode" and get this same white-on-black text screen by holding down certain keys when the computer first turns on. it wont say "hello" like our example here, the text it displays will be much more formidable.

the most important thing to learn about this white-on-black text screen, is that **you can give computer commands any name you want.** until there were icons to click on, using a computer meant memorising a few basic commands.

if the goal is literacy, it helps immensely to actually understand the essence of computing, rather than just training yourself how to do very specific tasks on very specific systems. if you learn some of the fundamentals, you can stop training at the surface level (the interface) and become an empowered and intuitive user.

one of the most daunting things about learning computers is that people feel they have to learn too much at once. we can blame that on education, which shoves lessons into people half a textbook at a time, instead of a more natural, more gradual way of learning. more people learns computers as needed, even if you take a class, and at home you can set your own pace. learning one or two tricks at a time is the easy way to gain proficiency with computers.

you dont have to memorise everything, and you dont have to write down every step (unless youre sure it will help) for the same reason that most people dont have instructions on opening a door. you need to know the steps to opening a door, and theyre complicated if you write them all down, but with enough practice people can operate doors without a great deal of thought-- it just becomes second nature.

coding, of course is the process of turning second nature back into steps. but even with this, practice makes it become a more automatic (if creative) process.

commands are just the names of computer programs, the text-only version of an icon. the most important things the computer names fall into three categories:

1. the program files, which do things
2. the data files, which dont do much
3. the names of devices, such as your dvd drive

you can run a program file by clicking its icon, or typing its name. you can open a data file the same way. devices, such as drives, often have icons as well. these things have names on the computer. one of the easiest commands to learn just lists files:

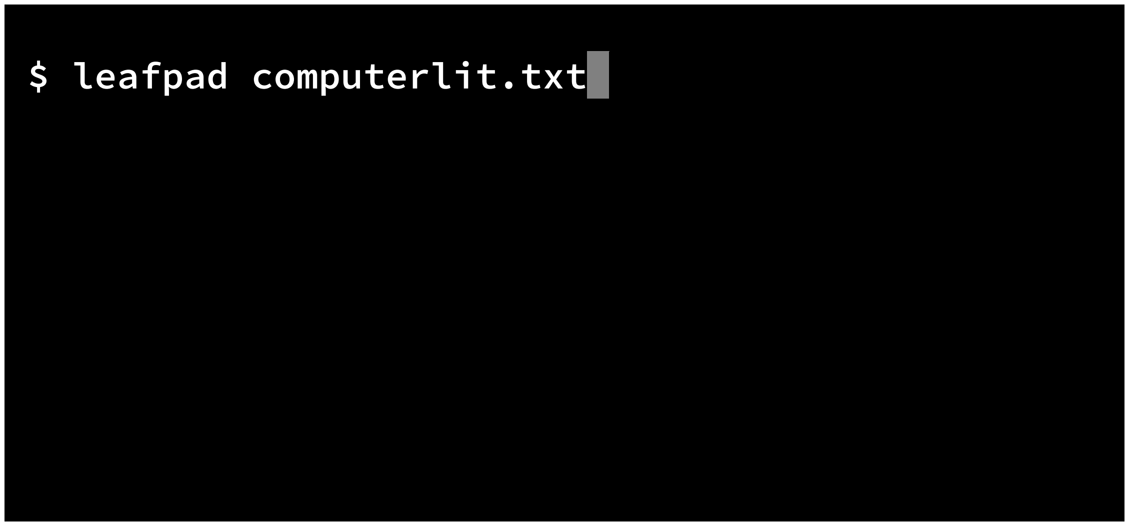
**dir**

if you type that command and press enter, it will either list the files in the current folder, or give you an error message. if it gives you an error message, you probably dont have a dir command (by that name at least.) on a mac or a gnu system, the command is called **ls**. but even some of these systems include **dir** just because people expect it.

**dir** and **ls** do pretty much what your web browser does if you open a folder on your computer with it. in the modern graphical version of everything, we use the term "folder" as a logical group of files. in previous decades it was known as a **dir**ectory.

once you have listed the files, you can tell the computer to open them. for programs, this means typing the name and hitting enter. for data, this means typing the name of the file as well as the program you want to open it with.

sometimes, the computer has a way of knowing the default program to open a file, so you only need to specify which file to open.



```
$ leafpad computerlit.txt
```

**open a file named computerlit.txt, using leafpad**

as it happens, leafpad is graphical program. you can open both text-based programs and graphical ones using the command line.

notice the dollar sign? you will often find a dollar sign in command line tutorials. it represents a prompt, and tells you that it is waiting for text on that line. the grey █ cursor also tells you exactly where the next character will be typed. sometimes it is only an underline, _ and sometimes it blinks to make it easier to spot.

maybe you dont want to type "leafpad" to edit a text file, maybe you prefer to say "edit" instead. first you would make certain there isnt already an edit command-- just type edit, hit enter, and if there is no edit command you will get a message like "**bad command**" or "**command not found**".  this is just the computer telling you it doesnt know what "edit" means.

depending on what operating system you use, you can change "leafpad" to "edit" in a variety of ways:

1. you could find the **leafpad** file and copy it to a file named **edit** in the same folder. if its real name is **leafpad.exe** or **leafpad.bat** you will want to change it to **edit.exe** or **edit.bat** instead. this only applies to windows users. using copying for this works, but is not ideal because when you update leafpad, you want the copy of "edit" updated too.

2. you could rename the leafpad file, but this has all the disadvantages of copying, plus it could break anything that is designed to look for leafpad. this is generally a bad idea-- its fixable of course, but you probably dont want to do it.

3. creating a shortcut using **ln -s** (dont worry, youll learn how if you want to) or creating a .lnk file in windows is probably the best way to do this. if you cant figure that out, copying is ok. understand that copying isnt a habit you want to get into, because if a security update changes the original leafpad, then leafpad will be safer but your old copy wont get the update (youll have to update the copy by copying the new one.)

now you can say **edit computerlit.txt** at the prompt. and youve just made your own command!

of course, its really just a second name for the leafpad program. but thats ok, if you want to get a program like leafpad and modify it into your own version, you can do that too. its just a lot more work whan all you wanted to do today is change the name you use to run the program.

weve learned very little about how to discover new programs, nothing about how to actually create them yet, and so far all we know is how to run a program that already exists on the system (and rename it.) but you can also use text to install completely new programs from a list of thousands of free software titles. you can even use text to install a windows-like graphical environment (if you dont have one already.) we will learn about that in the next chapter.